Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1987-12

Memory efficient of nonlinear stocastic equations and C applications

Connell, John C., Jr.

http://hdl.handle.net/10945/22351

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

LIBRARY
GRADUATE SCHOOL

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

C 7045

MEMORY EFFICIENT EVALUATION OF NONLINEAR STOCHASTIC EQUATIONS AND C^3 APPLICATIONS

by

John C. Connell, Jr.

December 1987

Thesis Advisor: Lester Ingber

Approved for public release; distribution is unlimited.

T238776

URITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited
DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION aval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 61	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
ADDRESS (City, State, and ZIP Code) onterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

TITLE (Include Security Classification)
Memory Efficient Evaluation of Nonlinear Stochastic Equations and C^3 Applications

PERSONAL AUTHOR(S)
Connell, John C., Jr.

TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987, December	70

SUPPLEMENTARY NOTATION

COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Command, Control & Communications, Monte Carlo Simulation, Nonlinear Probability Distributions, Neural Computers, Computer Simulation & Modeling

ABSTRACT (Continue on reverse if necessary and identify by block number)

The Statistical Mechanical Neural Computer (SMNC) developed in this thesis utilizes a Statistical Mechanical Nonlinear Algorithm (SMNA) to determine the long-time probability distribution of highly nonlinear stochastic systems. The use of the SMNA and a novel mesoscopic scaling technique help provide the SMNC with the capabilities of neural computers without the drawbacks of huge connection matrices and their attendant computational requirements.

In this thesis, the SMNC is initially used to verify the ability of the SMNA to duplicate relatively simple, single variable path integral solutions to nonlinear Fokker–Planck equations. After the fundamental algorithms are validated, the SMNC's ability to simulate a two-variable, multicellular problem by modeling a portion of the neocortex consisting of 10^5 neural units is discussed.

There are many important applications of the SMNC and its unique SMNA to C^3 systems including radar, sonar and electronic signals processing, missile guidance systems and an integrated battle management system. Such C^3 systems will benefit from the SMNC'S potential to efficiently filter large amounts of data, recognize patterns and anticipate, with some degree of uncertainty, the future state of highly nonlinear stochastic systems.

DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Lester Ingber	22b. TELEPHONE (Include Area Code) (408) 646-2687 22c. OFFICE SYMBOL 6111

FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-243

Approved for public release; distribution is unlimited.

Memory Efficient Evaluation of Nonlinear Stochastic Equations And C^3 Applications

by

John C. Connell, Jr.
Commander, United States Navy
B.S., University of Tennessee, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

December 1987

ABSTRACT

The Statistical Mechanical Neural Computer (SMNC) developed in this thesis utilizes a Statistical Mechanical Nonlinear Algorithm (SMNA) to determine the long-time probability distribution of highly nonlinear stochastic systems. The use of the SMNA and a novel mesoscopic scaling technique help provide the SMNC with the capabilities of neural computers without the drawbacks of huge connection matrices and their attendant computational requirements.

In this thesis, the SMNC is initially used to verify the ability of the SMNA to duplicate relatively simple, single variable path integral solutions to nonlinear Fokker–Planck equations. After the fundamental algorithms are validated, the SMNC's ability to simulate a two-variable, multicellular problem by modeling a portion of the neocortex consisting of 10^5 neural units is discussed.

There are many important applications of the SMNC and its unique SMNA to C^3 systems including radar, sonar and electronic signals processing, missile guidance systems and an integrated battle management system. Such C^3 systems will benefit from the SMNC'S potential to efficiently filter large amounts of data, recognize patterns and anticipate, with some degree of uncertainty, the future state of highly nonlinear stochastic systems.

Thesis
C7045
c.1

TABLE OF CONTENTS

I. INTRODUCTION 6

II. AN INTRODUCTION TO NEURAL COMPUTERS 10

A. THE BRAIN 11

B. EARLY NEURAL COMPUTERS 12

C. DEVELOPMENTS IN NEURAL COMPUTERS 14

III. THE STATISTICAL MECHANICAL NONLINEAR ALGORITHM 17

A. THE SMNA 18

B. VERIFICATION OF THE SMNA 21

C. VARIATION OF PARAMETERS 24

IV. SMNC THEORY OF OPERATION 31

A. TERMINOLOGY AND NOTATION 31

B. THE MACROSCOPIC SCALE 32

C. THE MICROSCOPIC SCALE 32

1. Microscopic Parameters and Variables 33

2. Microscopic Interactions 34

3. Microscopic Parametric Values 35

D. THE MESOSCOPIC SCALE 36

1. Mesoscopic Parameters and Variables 37

2. Mesoscopic Interactions 38

3. Mesoscopic Parametric Values 40

4. Mesoscopic Variables 41

V. CONCLUSIONS AND RECOMMENDATIONS 43

A. THE NEURAL COMPUTER 43

B. THE SMNA 44

C. THE SMNC 45

D. RECOMMENDATIONS FOR FURTHER RESEARCH 46

1. The Neocortex Simulation 46

2. JANUS 47

3. The Battle Manager 49

APPENDIX A: STOCHASTIC ROUTINES 51

APPENDIX B: SINGLE VARIABLE CODE 55

APPENDIX C: THE CENTERING MECHANISM 60

LIST OF REFERENCES 62

INITIAL DISTRIBUTION LIST 66

I. INTRODUCTION

Military tacticians have traditionally studied historical accounts of battles in order to gain an intuitive appreciation for the effects of variables such as force size, geographic positioning and weapons capability on the outcome of battles. These studies are conducted in the hope of gaining some tactical advantage over enemy forces in future battles. The study of simulated battles, that is military exercises, enhances this effort since battle variables can be controlled individually in repeated "engagements" and thus their effect on the outcome of the "battle" can be determined more precisely. Unfortunately, battles (and exercises) are highly stochastic events with a large number of variables, and significant analysis requires a large number of repetitions in order to produce statistically significant results. Further, modern military exercises involving large quantities of men and equipment are expensive and time consuming. At present there is just not enough battle data, real or exercise, available to satisfy the needs of analysts.

War gaming and computer simulation of large-scale combat scenarios would seem to present a viable solution to this dilemma given the recent technological breakthroughs in computer science. But the level of acceptance of computer simulations in major military battle management and procurement decisions appears to be low. Military planners need the capability to judge the ability of a computer simulation or model to accurately replicate actual, or at least exercise, battle data. Once a reasonable confidence level in computer simulations is obtained, then the simulations can generate data for force level or procurement decisions. One example might be a sensitivity analysis of sets of

simulation data where specific weapons characteristics are varied. Such analysis could prove to be extremely valuable in assigning proper weights to the differing characteristics and determining their proper influence in the context of a full-scale battle scenario.

This research project involves the development of a statistical mechanical neural computer (SMNC) which incorporates a Cauchy-driven Monte Carlo path-integral algorithm for calculating nonlinear long-time probability distributions. Although perhaps not immediately obvious, this SMNC offers a wide range of potential applications in the field of military command control and communications (C^3) such as

- validation of computer simulated combat data,
- efficient filtering of large amounts of data,
- pattern recognition and learning.

A computer system with these features will certainly be of great utility in the Navy's (C^3) systems. A data filtering system could provide complete and rapid analysis of acoustic, radar, or electronic warfare data. Pattern recognition and learning systems might be used as part of a cruise missile seeker to provide an enhanced target discrimination capability. A system that combines several of these capabilities could form the core of a battle management system. It would analyze sensor and communication data, recognize developing battle scenarios, anticipate the most likely enemy actions, and suggest the courses of action most likely to favorably affect the outcome of the battle.

The statistical mechanical neural computer presented here is a crucial step in the process of building such a realtime computer system. The SMNC is a virtual computer

running in the C programming language on a VAX 11/785 operating under the UNIX™ operating system. The choice of physical computer, programming language and operating system with which to implement the SMNC was made based on equipment availability at the Naval Postgraduate School and the portability of the program to others working in the field.

The SMNC incorporates a novel Statistical Mechanical Nonlinear Algorithm (SMNA) set forth by Ingber in a series of papers on the Statistical Mechanics of Neocortical Interactions (SMNI) [1-4] that reduces computational requirements to orders of magnitude below those of more traditional neural network computers.

According to Haken [5], the solution of nonlinear Fokker–Planck equations can lead to understanding of the state of variables in a complex system some time after the state of the variables is known. The ability to generate a long-time probability distribution for variables in nonlinear systems can provide valuable insight in many fields of science including chemistry, fluid dynamics, biology, thermodynamics, and even economics. Nonlinear Fokker–Planck equations are difficult to evaluate for all but the simplest systems. Nevertheless, Wehner and Wolfer have set forth a technique for numerical evaluation of path integral solutions for Fokker–Planck equations that has produced excellent results [6].

The SMNA is a memory efficient algorithm that quickly finds an approximation of the long-time probability distribution of nonlinear systems given an expression of a short-time probability distribution called a Lagrangian. Lagrangians can be fit to combat data such as that collected from repeated runs of JANUS computer simulated battles.

UNIX is a trademark of AT&T

Once the Lagrangian is found by a method like the one described by Upton [7], the SMNC using the SMNA can, with some degree of uncertainty, predict the outcome of a battle given the initial state of the system in terms of red and blue force compositions and locations. The SMNA is capable of modeling multi-variable systems as well as systems whose variables are influenced by what happens in other spatial grids.

Each year, the Office of Naval Research promulgates a list of priority research items [8]. This thesis in concerned with two items on the current list; modeling of heterogeneous, multivariable systems, and studies of neural architectures.

Chapter II serves as a background chapter and explains the history and operation of neural computing systems. This chapter also provides an overview of current work in neural information processing. Chapter III explains the operation of the statistical mechanical nonlinear algorithm (SMNA). The SMNA is the core of the computational efficiency of the SMNC. Once the algorithm is described, its operation is tested against nonlinear functions whose solutions have been determined by other means. Chapter III deals exclusively with single variable systems where the influence of neighboring "cells" has no effect on the state of the variable. Chapter IV contains a mathematical description of the statistical mechanical neural computer (SMNC) as used in a two-variable multi-spatial simulation of the operation of the neocortex portion of the brain. This chapter also explains the scaling methods used by the SMNC and how the microscopic scale can be used to provide feedback and control to the mesoscopic or middle scale. The summary, conclusions and recommendations for future work are presented in Chapter V.

II. AN INTRODUCTION TO NEURAL COMPUTERS

In late 1986 and early 1987 the popular press [9-15] carried cover articles heralding neural net computers as a significant step in the creation of thinking machines. Neural computers work by imitating the simultaneous interactions of the many neurons in the brain of a living animal and are based on concepts originally set forth in the 1940's.

Early attempts to emulate the operation of the brain produced largely disappointing results because of the widespread acceptance of faulty assumptions and the primitive computers available at the time. In the 1960's, when Massachusetts Institute of Technology's Marvin Minsky [16] and others criticized the concept on the grounds of insufficient knowledge of the brain, funds evaporated and research declined. Since 1970 there has been renewed interest in neural networks [17, 18]. The fruits of the renewed interest are to be found in the dramatic results described in the popular press articles.

The human brain is composed of about 10^{10} neurons each receiving information from about 10^{4} neighboring neurons and sending information in turn to 10^{4} of its fellows [19]. Since a network of only 200 simulated neurons can learn to read and talk within a few hours [11], the computational and learning ability of 10 billion interconnected neurons staggers the imagination. If shear numbers of neurons and their interconnections provide stumbling blocks in an attempt to model a human brain, the wealth of information available about the neuron itself at least provides some hope that the neuron can be successfully modeled and smaller but nevertheless useful neural computers constructed.

A. THE BRAIN

A neuron is a specialized brain cell consisting of a nucleus, many fibrous dendrite branches and a fibrous axon as in Figure 2.1 [14]. The dendrites are short tree-like extensions of the cell body that receive afferent (incoming) signals from the longer, thinner axons of other neurons. Neurons possess two characteristics not shared by other cells in the human body, they are excitable and have process. Excitability means that a neuron responds to certain stimuli with specific activity, and process permits the neuron to transmit efferent (outgoing) signals over distance [20]. These properties are fundamental to a neuron's information processing ability. Signals are chemically transmitted from one neuron to another across a synaptic cleft between an axon and dendrite. The signals received from the axons of neighboring neurons via the dendrites are integrated in the cell body. This integration process then determines whether the neuron will fire a signal to other neurons.

About 65% of a neuron's synapses are excitatory and encourage neural firing, while others are inhibitory and exhibit an inclination to cancel the effect of excitatory synaptic inputs [2]. Simply speaking, the neuron electrochemically sums all the excitatory and inhibitory signals received over a relaxation time of about 5 to 10 milliseconds. If the summation of these signals exceeds a pre-determined internal threshold value, the neuron fires a short duration, spike-like electric signal of about 100 mV down its axon toward the dendrites of its neighbors. The output signal is non-linear in that its strength is independent of both the threshold value and the integrated value of the inputs. At any given time, a neuron is in one of two states: firing or not firing [21]. However, the rate of firing may increase in response to sustained excitatory signals. Complications arise given

the likelihood that a neuron can modify its threshold in response to activity [21] or that a particular synapse can experience augmentation or diminution of strength relative to other synapses in a particular neuron [22].

This operation of neurons is a complex and sensitive interaction of chemical and electrical processes involving a vast number of neurons and other brain cells. Of course it is not sufficient to model a neuron without also considering its thousands of synapses. The problem is no longer one of simulating 10^{10} neurons, but also 10^{14} or so synapses [19]. The huge connection matrix that accompanies each attempt to build a neural computer has proven to be a major stumbling block and attempts to deal with the problem have met with varying degrees of success.

B. EARLY NEURAL COMPUTERS

Hawkins [23] mentions three properties of neurons which were generally accepted by early experimenters in neural computers:

- Synchronous operation,
- Binary operation, and
- Linear, weighted summation of inputs.

Synchronous operation was necessitated by the need to deal with discrete time intervals both mathematically and in digital computer simulations. Partial justification was offered that since long neurons transmitted at higher speeds than shorter ones, the delay time in a given region of the brain tended toward a constant of about 5 msecs. More recently, however, Peretto and Niez [24] argue for asynchronous neural activity, since signals arrive at synapses at irregular times.

Binary operation, the all-or-nothing neural firing, has been experimentally observed and is an easily modeled characteristic, especially on a digital machine. But again, current research by Hopfield [25] suggests a more useful model is achieved with graded-response neural units.

Linear weighting of neural inputs provided a simplifying assumption for mathematical convenience, but SMNI [1-3] shows that a non-linear, non-equilibrium Lagrangian function more rigorously describes the summation of inputs.

Given the widespread acceptance of these assumptions and the quality of computers available 30 years ago, it is amazing that scientists achieved the results they did. According to Hawkins [23], Rashevsky was the first to attempt to mathematically describe biological processes in 1938. McCulloch and Pitts [24] pointed out the utility of applying Boolean algebra to neural nets in 1943 and developed a neural model that served as a foundation for much of the early work in the field. Rosenblatt [22] achieved a high water mark with his work on the perceptron in the late 1950's.

Rosenblatt's mathematical perceptron model consists of a network of sensory units, association units and response units. Sensory units respond to external stimuli by emitting a signal which is some function of the input energy. The association units generate output signals if the algebraic sum of the input signals exceed some threshold value. Finally, the response unit transmits a signal outside the network if the sum of its inputs is positive.

After thorough and exhaustive conceptual work with perceptrons constructed with a variety of architectures, Rosenblatt reported "With a suitable design and training procedure, a three-layer series-coupled perceptron can be taught to duplicate the

performance of any finite automaton." Since a finite automaton is considered to be the fundamental unit of computation, Rosenblatt in effect claimed that the only limits to the ability of a network to learn input-output relationships are the size, speed, reliability and complexity of the network. In 1960, a hardware model of the perceptron featuring a 20 x 20 array of association and response units verified Rosenblatt's network learning concepts and successfully learned to recognize letters of the alphabet [23].

Minsky and Papert [16] criticized the perceptron, asserting that too little is known of the human brain to provide justification for any attempt to model it. Considering Minsky's stature in artificial intelligence, it is little wonder that funding abruptly disappeared for work in neural networks and neural computers.

C. DEVELOPMENTS IN NEURAL COMPUTERS

According to Hecht-Nielsen [26], the goal of artificial neural systems research is to create man-made systems that can process the kind of information that brains process. Examples include

- real-time high-performance pattern recognition,
- knowledge processing for inexact knowledge domains, and
- fast, accurate control of robotic movement.

He thinks the solution to these problems lies in providing a network of neural units with the capability to self-adjust their responses to experience. Port [11] reports that such a capability would permit a neural computer to process analog rather than digital signals, make weighted decisions based on "fuzzy" data, find close matches in large data banks, and compute fast but approximate answers to time-consuming problems.

Neural network simulation is difficult on serial digital computers because of the huge connection matrix that must be updated with each tick of the system clock. Statistical sampling techniques can remove some of the computational burden, but only to a limited degree if reliability is to be maintained. One solution seems to lie in massive parallel processing systems such as the Connection Machine [27] and the Transputer. [28] Another approach has been to make many calculations in the shortest time possible on supercomputers like Cray-2. Hecht-Nielsen, and IBM's Cruz-Young [26] have designed parallel processing add-ons for IBM PC's to reduce the expense of neural network research yet maintain speed and reliability.

The statistical mechanical neural computer (SMNC) provides a tremendous reduction in computational requirements without an attendant loss in capability through the introduction of a mesoscopic scaling technique. Although the SMNC is implemented on a serial digital computer to demonstrate the mesoscopic algorithms, it could be implemented on a parallel processing system. The combination of parallel processing and the reduced computational load provided by the mesoscopic scaling algorithms would provide a real-time neural network system.

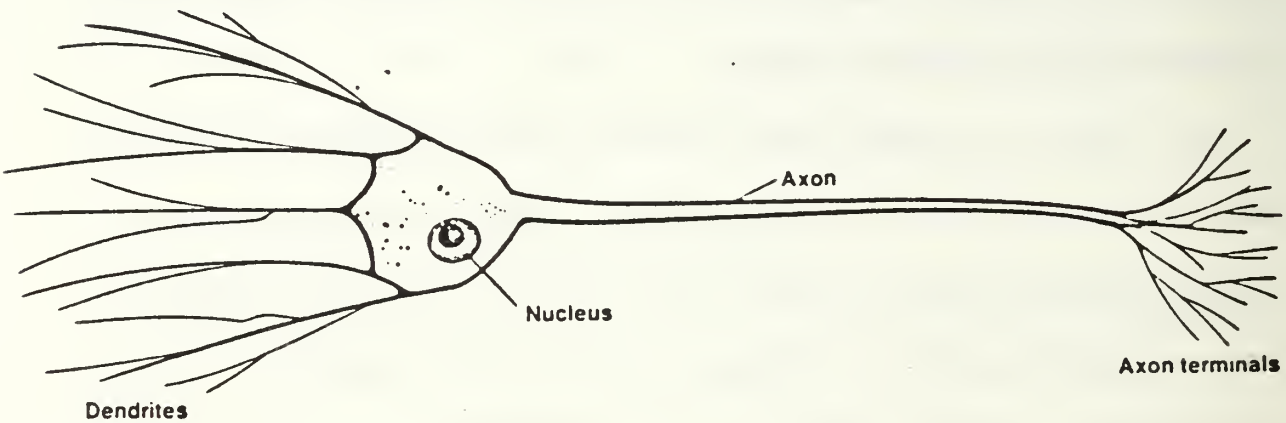

Figure 2.1 A Typical Neuron

III. THE STATISTICAL MECHANICAL NONLINEAR ALGORITHM

This chapter develops a method for determining long term probability distributions for nonlinear nonequilibrium systems using a combination of modern statistical mechanical modeling techniques. The method combines simulated annealing, Monte Carlo techniques and a method for simulating path integrals as developed in a series of papers dealing with the "Statistical Mechanics of Neocortical Interactions" (SMNI) [1-4].

This new method for calculating long term probability distributions has great potential in the area of Navy Command Control and Communications (C^3) Systems. Such a system might help to forecast the position of friendly and enemy tanks or ships given some understanding of the deterministic and stochastic forces that influence their positioning during battle. For example, some of the factors that might affect the position of a ship in battle are maximum speed capability, ammunition and fuel capacity, operational orders, weather, geography and the disposition and activity of enemy ships, submarines and aircraft. Ingber and Upton have developed a method for examining battle scenarios and translating the deterministic and stochastic forces into mathematical expressions for short-time probability distributions [29].

The algorithm developed in this chapter uses such mathematical expressions, called Lagrangians, to generate nonlinear long-time probability distributions. The algorithm used throughout the thesis is referred to as the Statistical Mechanical Nonlinear Algorithm (SMNA) for convenience.

Combat is a highly stochastic highly nonlinear event in which opponents operate at extreme rather than average capabilities. Thus a quasi-linear deterministic model of

combat such as Lanchester theory has difficulty modeling any real combat data with any reasonable degree of precision [30]. The only hope of understanding such a complex nonlinear system lies in the aggregation of many trajectories in time and space from each scenario in which one parameter has been varied. The sensitivity of the system to the change of a parameter can then be quantified and used in decisions involving the use of the parameter in different combat scenarios. The SMNA provides the capability to aggregate many trajectories in time and space, and repeated operation with individually varying parameters can result in truly meaningful decision aids.

A. THE SMNA

The Boltzmann method of rejection test as described by Schulman [31] and Metropolis et al [32] is the heart of the statistical mechanics in the SMNA. According to Landau and Alben [33], who built on the work of Metropolis et al, the state designation of a system after many cycles may be found by beginning with any state designation and generating enough successive designations such that the probability of any designation is given by the Boltzmann distribution. In general, the SMNA engine steps through time and space assigning values to system variables at each step. A neural computer as described in Chapter II is an ideal vehicle for this nonlinear long-time probability algorithm since each neural unit integrates the effects of its spatial and temporal neighbors as it sets the values of its variables.

Consider a situation in which the initial value of a variable q and a mathematical description of the short-term probability distribution for temporal or spatial displacements of q are known. According to Haken [5], such situations arise in diverse areas of study such as chemistry, mechanics, optics, fluid dynamics, biology and

economics. A variable's short-time probability distribution may be found readily for adjacent temporal cells. However, long-time probability distributions can frequently be found only through solution of Fokker–Planck equations or their representation as path integrals. These path integrals are derived from a system of stochastic rate equations referred to as Langevin equations [5]. Exact solution of these equations can often be difficult to achieve in any but the simplest of systems.

The SMNA can be used to determine the approximate long-term probability distribution of q after time t. This is accomplished by calculating many trajectories that q might follow through space and time, and examining the aggregated values of q after the simulation of these many trajectories. An aggregation of the final values of q will reveal the form of a long-time probability distribution from the short-time function under examination. This process is discussed for the case of a single variable occupying a single spatial grid cell, but cycling from time t_0 to time t in steps of size Δt.

In the case of a such a single variable system that varies in time but not in space, the Boltzmann probability distribution can be written as

$$P = (2\pi Q(t-\Delta t)\Delta t)^{-1/2} e^{-L\Delta t} \tag{3.1}$$

where

$$L = \frac{(q(t)-q(t-\Delta t)-K(t-\Delta t)\Delta t)^2}{2Q(t-\Delta t)\Delta t^2}, \tag{3.2}$$

and $K(t-\Delta t)$ is the drift force and $Q(t-\Delta t)$ is the diffusion coefficient. K and Q may be highly nonlinear functions of q. If L is a function of one variable, say q, then it is a function of both $q(t)$ and $q(t+\Delta t)$. These variables can be referred to as q and q'.

If the value of q is determined for each time t_s as s is incremented from 0 to N, where $N = t/\Delta t$, it may be plotted versus t to represent a *trajectory* through time. Assuming that q is known for all time steps s in the previous trajectory $(n-1)$ and that $L(q,q',t)$ is known for the current trajectory, n, then $L(q,q',t+\Delta t)$ may be found for trajectory n. In order to find $L(q,q',t+\Delta t)$ for trajectory n, a value for q' must be selected and used to calculate a trial value for $L(q,q',t+\Delta t)$. A stochastic process is used to sample the space and select a value for q' and the Boltzmann method ensures that only likely values of q' are accepted as q's and unlikely values are rejected.

The Cauchy routine described in Appendix A is called with q from trajectory $n-1$ as the mean and an experimentally determined value for the temperature. The use of the Cauchy routine with a variable temperature provides an effect similar to the "fast" simulated annealing process described by Szu [34]. The Cauchy routine samples the variable space "near" the value of q in the previous trajectory, but the "fat tail" of the Cauchy distribution ensures that the entire space is sampled, and if multiple system minima exist, they can be found. This use of the Cauchy routine differs from traditional Monte Carlo methods which only sample the variable space in a very narrow, uniformly distributed band centered on the value of q in the previous trajectory.

The value q' returned by the Cauchy routine is entered in Eq. 3.2 for trajectory n, time steps s and $s+1$. Then DL is determined from

$$DL = L_n((s+1)\,\Delta t) + L_n(s\ \Delta t) - L_{n-1}((s+1)\,\Delta t) - L_{n-1}(s\ \Delta t)\,. \tag{3.3}$$

Only $q_{n-1}(s\,\Delta t)$ is changed to $q'_n(s\,\Delta t)$. Again, the subscript n indicates trajectory and subscript s indicates time step within the trajectory. Thus, DL describes a perturbation of the entire $(n-1)$th trajectory at time $(s\,\Delta t)$. This is done for all s.

DL is now used as a basis for the acceptance or rejection of q'. A random variant x, uniformly distributed on (0,1), is generated and if

$$x < e^{(-DL)(\Delta t)}, \tag{3.4}$$

then q' becomes q for trajectory n step s. The process is repeated until time t is reached when the value of q is captured and plotted on a histogram. After a large number of trajectories, the histogram may be examined to reveal the long-term probability distribution of the functions under examination.

This procedure may be further generalized to represent systems in which the value of a variable is influenced by variables in neighboring spatial cells and systems of more than one variable. The Statistical Mechanical Neural Computer (SMNC) described in Chapter IV describes the potential use of the SMNA on a system of two variables and 1089 spatial cells. The JANUS tank battle simulation is a two variable system involving a small spatial grid of 9 cells. The SMNA is sufficiently general to possess applications in many areas such as nuclear physics, neurobiology, chemistry, metallurgy, economics and military C^3.

The multivariable multicellular Lagrangian as described by Ingber [35] forms the basis of the SMNC equations of Chapter IV.

B. VERIFICATION OF THE SMNA

Appendix A discusses in detail the steps taken to ensure that the stochastic routines for the SMNA were properly written and that the algorithms performed as expected. Verification of the overall operation of the algorithm is difficult since, in complex nonlinear cases, the long term probability distributions cannot be determined empirically.

Without an empirical "right answer" against which to measure the SMNA no verification is possible.

Wehner and Wolfer faced a similar problem with their path-integral method for finding long term solutions to various distributions [6]. They verified their method of evaluating path integral solutions to Fokker–Planck equations by trying it on relatively simple nonlinear functions whose solutions are known. To verify the efficacy of the SMNA, it was used to reproduce the results achieved by Wehner and Wolfer.

Initially, they chose a Lagrangian function used in the Rayleigh gas model of a dilute concentration of heavy atoms in a gas of light atoms [6]. The Fokker-Planck equations representing this system has a drift force $K(q) = -q + 3/2$ and a diffusion coefficient $Q = q$. The long term solution is known to be

$$P(q,t) = \frac{e^{t/2}}{2\pi q_0 (1 - e^{-t})^{1/2}} \left[e^{\exp 1} - e^{\exp 2} \right], \tag{3.6}$$

where

$$\exp 1 = \frac{\left[q^{1/2} - (q_0 e^{-t})^{1/2} \right]^2}{1 - e^{-t}}, \tag{3.6a}$$

and

$$\exp 2 = \frac{\left[q^{1/2} + (q_0 e^{-t})^{1/2} \right]^2}{1 - e^{-t}}. \tag{3.6b}$$

Wehner and Wolfer's results are plotted in Figure 3.1a with $q_0 = 7.0$ $t = 0.5$ minutes, and Δt=0.1 min. The SMNA solution is plotted in Figure 3.1b with Δt=0.5 min. and all other parameters the same as Wehner and Wolfer's. Figures 3.2a and 3.2b show a similar comparison of results for time $t = 5.0$ minutes. The similarities between the two plots provides verification of the SMNC's ability to predict coarse grained long term probability distributions in the case of a relatively simple nonlinear function.

The SMNA finds a long term probability distribution that is similar to the exact solution but has done it in a manner that is simple enough to be implemented on a micro-computer yet general enough to be used in a variety of situations. This method is a variation of the traditional Monte Carlo method of Metropolis et al [32] with an added feature of using the Cauchy distribution, instead of a uniform distribution over a small range, to deal with nonlinear problems. The SMNA produces a solution with somewhat lower resolution than the Wehner and Wolfer method, but the solution was achieved in a manner of minutes on a VAX 11/785 and only required enough memory to support a one-dimensional data array of 3 floating point numbers in each element and N elements where $N = t/\Delta t$.

After reasonable success with a simple equation, the algorithm method was tested on a more complicated Fokker–Planck equation representing a bifurcating solution. Here $K(q) = \tanh q$ and $q = 1$. The long-time solution is

$$P(q,t) = \frac{\operatorname{sech}(q_0)}{(2\pi t)^{1/2}} e^{-t/2} e^{-(1/2t)(q-q_0)^2} \cosh(q) . \tag{3.7}$$

Figure 3.3a shows the Wehner and Wolfer results for $t = 10$ min, $\Delta t = 0.1$ min. and $q_0 = 0.0$. Figure 3.3b plots SMNA results with $\Delta t = 1.0$ min. Figures 3.4a and 3.4b are the

results of setting $q_0 = 0.6$ and dramatically illustrate the SMNA's sensitivity to initial conditions. The only concession to the bifurcation was the necessity to repeat the initialization process at intervals during the collection of data.

This exercise in reproducing Wehner and Wolfer's results from path-integral calculations with an entirely different approach using statistical mechanics provided confirmation that the SMNA is fully capable of finding long-time nonlinear probability distributions quickly and with a reasonable degree of resolution.

C. VARIATION OF PARAMETERS

During experimentation with the SMNA it became obvious that the algorithm was sensitive to many different program parameters. Appropriate values were chosen for each parameter based upon educated guesses, experimentation and mathematical deduction. The limited time and computer resources available to this research project combined to prohibit a proper sensitivity analysis of each parameter and its effect upon the operation of the algorithm, but over the course of literally hundreds of runs of the SMNA, a good guage for these parameters was developed. The following paragraphs describe, in general terms, the influence of these parameters.

The number of trajectories. The resolution of the graphic presentation of the data is inherently coarse since data points are represented by ASCII characters 1/10 inch wide and 1/6 inch high. However, the output plots were discovered to be unacceptably coarse when less than 10,000 trajectories were plotted. No improvement in resolution was noted above about 20,000 trajectories and consequently, 20,000 was used for the final runs.

The size of Δt. The mesh size Δt represents the size of each time slice in the integration process. In the discrete mathematics, a smaller Δt generally means increased accuracy and longer run times. The SMNA generally worked well when Δt was chosen to be about $1/L_{static}$ where L_{static} was the value of the Lagrangian with the q' term set to zero, that is, $L_{stat} = K^2/2Q$. In the Rayleigh Gas case Δt =0.5 min. and in the bifurcating process, Δt =1.0 min.

Cauchy temperature. The "temperature", or in a sense the variance, of the Cauchy routine had an effect on both the form of the results and the efficiency of the program. When the temperature was too high, many unsuitable values for q' were submitted for acceptance and then rejected, thereby slowing down the operation of the code. On the other hand, too narrow a temperature range produced a situation that failed to sample the entire variable space and consequently missed the location of the some of the multiple minima which occur in nonlinear systems. The temperature was chosen to be approximately the amount that q might reasonably be expected to change from one trajectory to the next. During the validation runs, the temperature was set to 1.0.

Warmup length. As the code begins operating, an initial trajectory is required. This trajectory can almost be chosen arbitrarily. The initial trajectory used was found by setting each q to the value of the initial conditions and then cycling through the code for several thousand cycles. This resulting trajectory represents a more "likely" trajectory than one composed of identical values of q and it serves as a suitable starting point for the SMNA code. The validation runs were "warmed up" by cycling the initial value trajectory through 2000 cycles, an amount determined though experimentation.

Warmup repetition. In order to achieve bifurcation within a reasonable number of runs, the trajectory initialization and warmup process was repeated every 1000 cycles. This reinitialization served to shortcut the stochastic nature of the SMNA by, in effect, reseeding the random number generator.

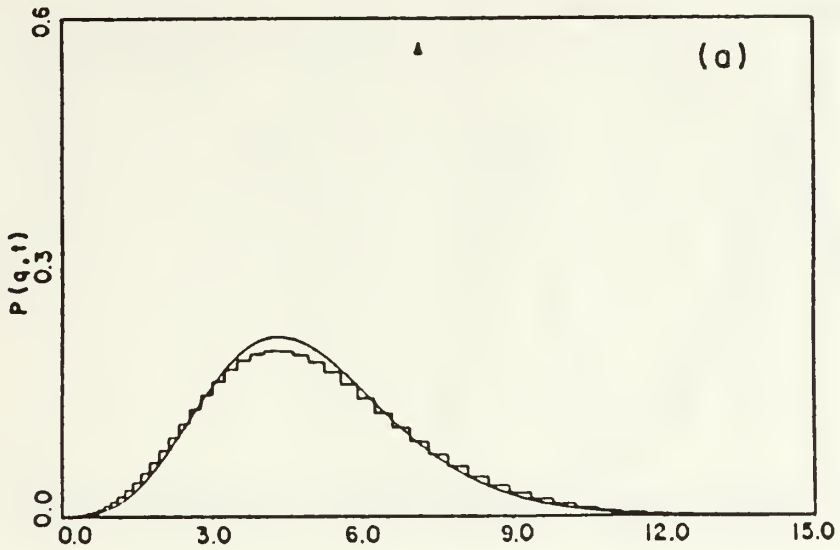

Figure 3.1a Wehner and Wolfer Plot of Rayleigh Gas System, $t = 0.5$ min.

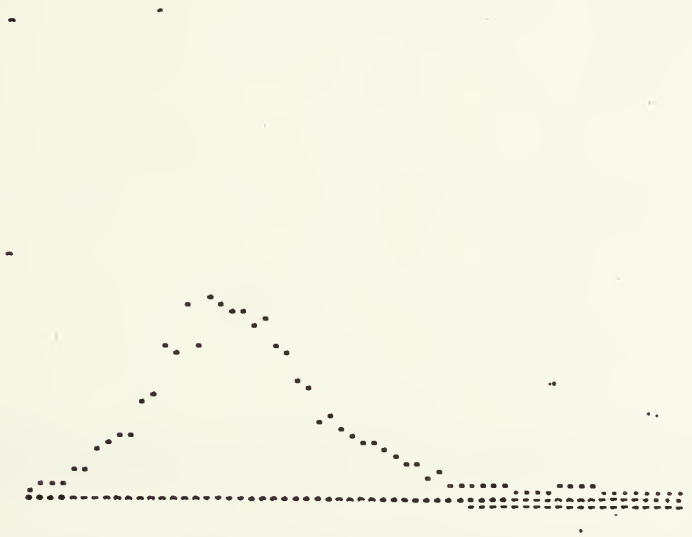

Figure 3.1b SMNA Plot of Rayleigh Gas System, $t = 0.5$ min.

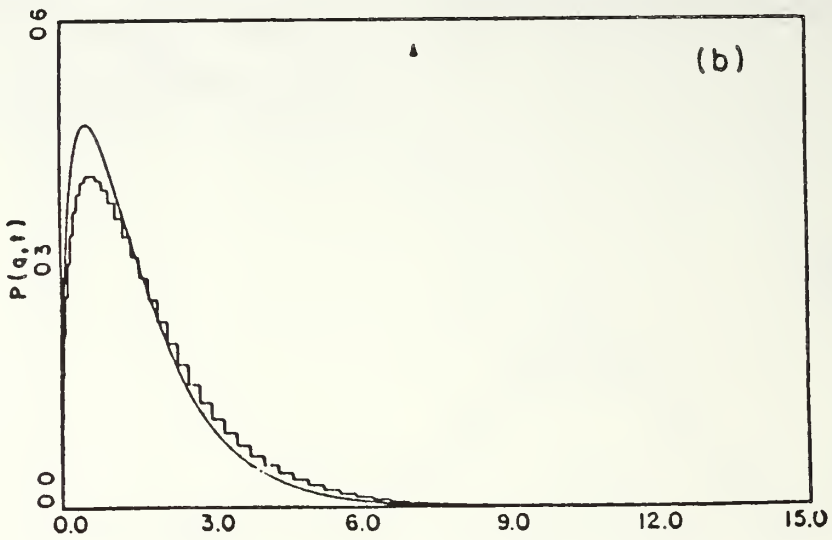

Figure 3.2a Wehner and Wolfer Plot of Rayleigh Gas System, t = 5.0 min.

Figure 3.2b SMNA Plot of Rayleigh Gas System, t = 5.0 min.

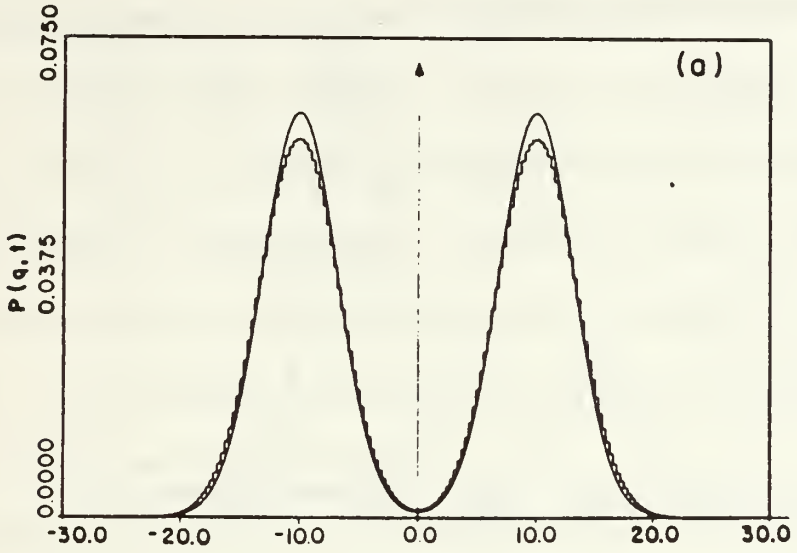

Figure 3.3a Wehner and Wolfer Plot of Bifurcating Process, $q_0 = 0.0$

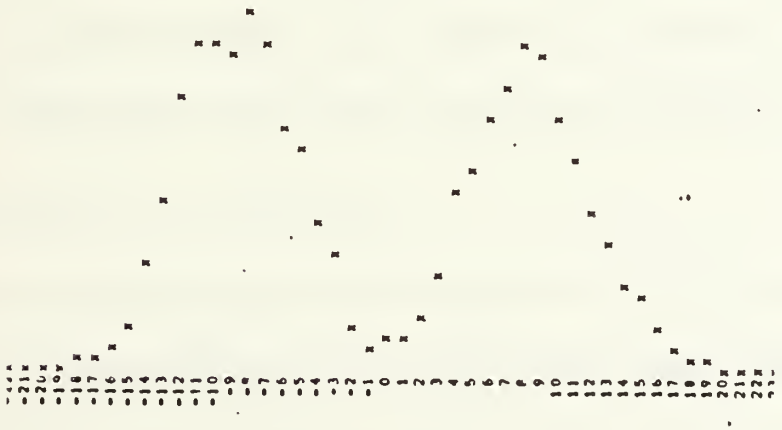

Figure 3.3b SMNA Plot of Bifurcating Process, $q_0 = 0.0$

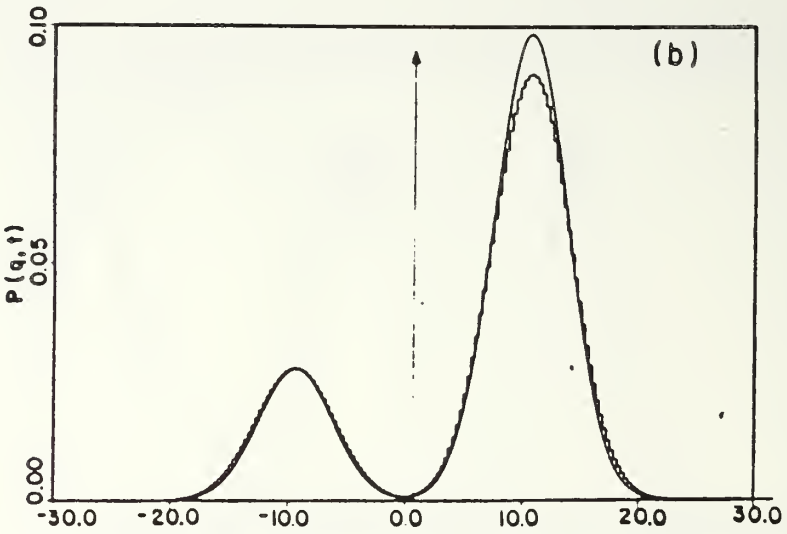

Figure 3.4a Wehner and Wolfer Plot of Bifurcating Process, $q_0 = 0.6$

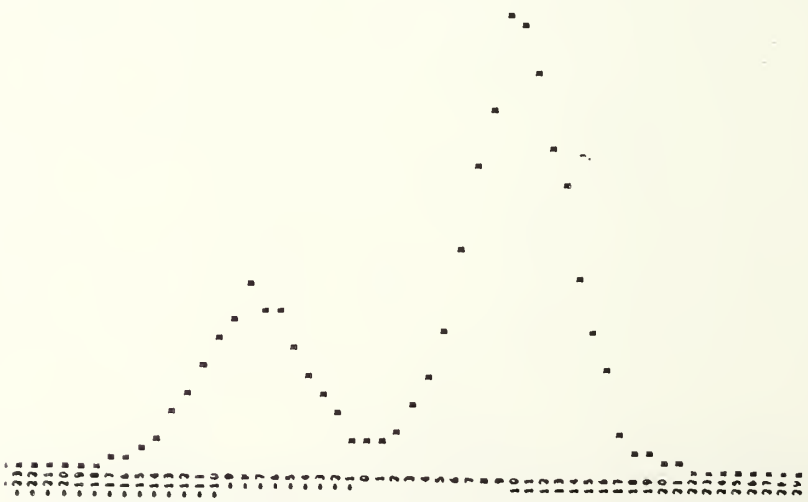

Figure 3.4b SMNA Plot of Bifurcating Process, $q_0 = 0.6$

IV. SMNC THEORY OF OPERATION

This chapter discusses the principles behind the operation of the two scales of the Statistical Mechanical Neural Computer (SMNC). One computer operates at the mesoscopic or middle scale and takes advantage of two statistical mechanical shortcuts that permit modeling of a neural computer consisting of 10^5 units. The other computer operates at the microscopic level and simulates a fully connected network of about 550 neural units. The microscopic network serves as a control and feedback mechanism for the more interesting mesoscopic computer.

The SMNC is modeled as closely as possible after the neocortex region of the human brain since much is known of that part of the brain. This modeling of the neocortex demonstrates the Statistical Mechanical Nonlinear Algorithm (SMNA) on the highly nonlinear multivariable multicellular system. The use of the SMNA and nearest neighbor algorithms highlights their utility in reducing the computational load normally associated with a neural computer. It also shows how the mesoscopic scale can be used to serve as an efficient filter between the microscopic and macroscopic scales.

A. TERMINOLOGY AND NOTATION

Since much of the important work of this thesis lies in C^3 applications, a general terminology and notation scheme is used and references to neurons, axons, dendrites and the like are severely limited. The neuron and synapse, the two chief entities in any model

---

Substantial portions of this chapter were published as: Connell, J., L. Ingber and C. Yost, "A statistical mechanical virtual neural computer and C3 Applications" In *Symposium on C3 Research.* National Defense University, Washington, DC. 1987.

of the brain are called *unit* and *link* in this thesis. In general, this chapter will follow the same notation conventions and use the same terminology as developed in a series of papers dealing with "Statistical Mechanics of Neocortical Interactions" (SMNI) [1-4].

B. THE MACROSCOPIC SCALE

The human brain contains 10^{10} neurons, and a model this large is impossible with today's technology. However, there are natural groupings of about 10^5 neurons in the neocortex that provide natural units for neural modeling. SMNI divides the neocortex into such groups of 10^5 units in an area of about $5\times10^9\ \mu m^2$ and calls them macrocolumns [2]. The physical placement of units within a macrocolumn with respect to one another is unimportant in this simulation; they may reside in a flat circle, in a sphere, or along a convoluted ribbon. However, the functional relationship of one unit to another is significant and must be preserved in the model. There is no notation convention unique to the macroscopic scale.

The mesoscopic SMNC models just one macrocolumn of 10^5 individual units. Although a macrocolumn is a small portion of the total human brain, its potential is staggering when compared with the number of units typically found in neural computers.

C. THE MICROSCOPIC SCALE

The microscopic scale is the level at which an individual unit communicates with its neighbors. Traditional neural net computers only consider interactions at this microscopic level. However, in the SMNC, the determination of individual unit firing states at the microscopic level is done in parallel with operations at the middle or mesoscopic scale. The microscopic level is used in the SMNC to:

- provide a frame of reference to the mesoscopic domain,
- demonstrate the influence of "commands" from the macroscopic level,
- send feedback to the mesoscopic level, and
- serve as a mechanism for adjusting mesoscopic parameters.

1. Microscopic Parameters and Variables

The parameters $A_{jk}, B_{jk}, v_{jk}, \phi_{jk},$ and V_j are used in the calculation of probability distributions which in turn determine the state of an individual unit's single variable σ_j. The subscripting convention has parameters related to individual units subscripted with j, where j ranges from 1 to 10^5, the number of units in a macrocolumn. A link and other parameters arising from an interconnection between a receiving unit j and a sending unit k are subscripted jk. The subscript k has the same range as j, but $j \neq k$.

A unit will form about 10^4 incoming, or *afferent* links with other units in its macrocolumn. In other words, a unit receives messages from ≈10% of its neighboring units and when it fires, it transmits to ≈10% of the units in the macrocolumn. The afferent signals received during the relaxation time τ, are summed by the unit. The results of the summation, p, are compared with the unit's threshold potential V_j and the unit most likely fires if V_j is exceeded. In the case of firing, the variable σ_j is set to 1, otherwise it is -1. The method of rejection test is used to determine whether or not a unit fires based on this comparison between p and V_j.

Two classes of units are required for the brain simulation: inhibitory (I) and excitatory (E). Inhibitory units send negative signals that discourage a unit from firing and excitatory units send positive signals. Since there are two classes of units, there are

four possible combinations of link interactions: E-E, E-I, I-I and I-E. Several of the microscopic parameters are sensitive to these combinations.

The parameter v_{jk} is the net electrical potential at the link between two units k and j during the firing of a signal from k to j. This potential is on the order of 0.1 mV and is positive if the signal was sent by an excitatory unit and negative if sent by an inhibitory unit. ϕ_{jk} is a parameter representing the statistical variance of this electrical potential.

A unit j assigns a weighting factors to each of its afferent links. This link weighting factor is denoted A_{jk}, and it may differ for each jk connection. The remaining parameter in the microscopic scale is B_{jk} which represents a random background noise factor which may differ for each jk link.

2. Microscopic Interactions

The firing of unit j is denoted by the state of the variable σ_j; σ_j=1 if j fires and σ_j=−1 if it does not. σ_j can be determined from a probability distribution p_{σ_j} which is derived by SMNI in [2].

$$p_{\sigma_j} = \frac{e^{-\sigma_j F_j}}{e^{F_j} + e^{-F_j}} \tag{4.1}$$

where

$$F_j = \frac{V_j - \sum_k (a_{jk} v_{jk})}{[\pi \sum_k a_{jk} (v_{jk}^2 + \phi_{jk}^2)]^{1/2}} \tag{4.2}$$

and

$$a_{jk} = \frac{1}{2} A_{jk}(\sigma_k + 1) + B_{jk} \ . \qquad (4.3)$$

summing over all the sending units k for each of the receiving units j. Once the SMNC calculates p_{σ_j}, σ_j is determined by the Boltzmann method of rejection as described in Appendix A [36].

3. Microscopic Parametric Values

Both v_{jk} and ϕ_{jk} are on the order of 0.1 mV and are found by calls to a Gaussian variant generating procedure. This procedure accepts a mean and a variance as input parameters and returns a variant with the properties of a Gaussian distribution. ϕ_{jk} is computed first with a mean of 0.1 and variance of 0.015. v_{jk} is then computed using 0.1 as mean and ϕ_{jk} as variance. Only positive numbers are initially accepted from the Gaussian procedure, but for inhibitory firings, I-E or I-I interactions, v_{jk} is multiplied by -1 to represent the negative potential.

Figure 4.1 contains values for the weighting factors A_{jk} and background noise factors B_{jk}. These parameters vary according to the excitatory/inhibitory (E,I) class identity of the sending and receiving units. The values in Figure 4.1 are found through statistical averaging techniques [2] at the mesoscopic scale discussed in the next section. Since the numbers in Figure 4.1a are mesoscopically scaled averages, they may not be directly substituted for A_{jk} and B_{jk}. These values are used as entering arguments to the Gaussian procedure to produce A_{jk} and B_{jk} values with correct distribution properties.

V_j represents a unit's threshold potential and serves as the yardstick against which the summation of signals is measured. A value for V_j is found with a call to the Gaussian generator using 10.0 mV as the mean input parameter and 1.5 as the variance.

In some areas of the brain, such as the cerebellum, the afferent and efferent interconnections are rigidly structured. However, in the neocortex, random processes determine both the number of connections and the specific neurons with which connections are established. In the SMNC, as in the brain, units establish links with about 10% of their neighbors. Whether or not a link exists between two units is determined in the initialization phase by calls to a uniform random number generator.

About 65% of the units in the brain model are excitatory. A call to a uniform random number generator initially establishes each unit as either excitatory or inhibitory, but the dynamic simulation permits units to change orientation as the simulation progresses.

The relaxation time τ is 5 to 10 msec in the neocortex. The SMNC operates synchronously with each cycle equal to $\Delta t \leq \tau$, although the neocortex almost certainly operates asynchronously. This can be justified by considering the following thought experiment: Suppose 100 people per hour pass a wishing well and 10% of the passersby throw in a coin. Surely the arrival of a coin in the well is asynchronous, but over the time span of a day or week, it could be said that about 10 coins arrived each hour. Thus nonsimultaneous microscopic events can be treated as simultaneous at the mesoscopic scale since they occur during the same time interval [36].

D. THE MESOSCOPIC SCALE

Haken [5] points out the need for a mesoscopic scale in nonequilibrium systems to formulate the statistical mechanics of the microscopic system. This formulation will permit development of the macroscopic scale and provide a means of filtering microscopic interactions as well as a channel for issuing "orders" in a C^3 application.

Further, the use of the SMNA and mesoscopic scaling dramatically reduces the computational burden traditionally associated with neural computers.

1. Mesoscopic Parameters and Variables

A minicolumn is a group of about 110 units which forms the basis for mesoscopic scaling. These groupings have been observed in the neocortex [37] and provide the key to the success of the SMNC. The parameters of the mesoscopic scale are $v_G^{G'}$, $A_G^{G'}$, and $B_G^{G'}$ which are minicolumnar-averaged synaptic parameters. The terminology used in the mesoscopic development is similar to that used for microscopic interactions.

The most significant difference in notation at the mesoscopic level is the use of superscript or subscript G where $G = E$ or I. Recalling that 65% of the 110 units in a minicolumn are excitatory, E may range in value from -80 to +80 and I may range from -30 to +30. A parameter with a single superscript G shows a summation over the E's and I's. A parameter with a superscript G and a subscript G' like $A_{G'}^{G}$, suggests a similar summation process over both the sending minicolumns, G', and the receiving minicolumns G. The variables of this system are M^G representing the two variables M^E and M^I. Thus, M^E contains information revealing how many of the excitatory units in a minicolumn fired during one time step. If the total number of excitatory units in the minicolumn is known, then the number of units that did not fire during the time step may be found easily.

The mesocolumns are functional groups closely associated with minicolumns. Conceptually, a mesocolumn represents an afferent minicolumn and an efferent macrocolumn; it is a statistical treatment of the signals sent and received by the 110 units

in a minicolumn. Since a macrocolumn contains about 10^5 units arranged in minicolumns, there are about 10^3 mesocolumns in each macrocolumn.

SMNI shows that it is not necessary to model every mesocolumn's interactions with every other mesocolumn [2]. The mesoscopic probability distribution can be found by computing the influence of a mesocolumn's four nearest neighbors rather than the $\approx 10^3$ other mesocolumns in the macrocolumn. In effect, a summation of about 4000 complex calculations replaces a summation of the 10^6 to 10^{10} simpler computations required to model 10^5 units and the resulting interconnection matrix in a more traditional neural computer.

2. Mesoscopic Interactions

The calculations necessary to produce a probability distribution representing the firing state of a minicolumn are more complex than those used for microscopic interactions and require the use of the Statistical Mechanical Nonlinear Algorithm (SMNA). SMNI establishes a mesoscopic Lagrangian L, representing the nonlinear short-time system probabilities [2]. The long-time, two variable probability distribution for the neocortical parameters is

$$P \approx \frac{1}{2\pi\Delta t} g^{1/2} e^{-N\Delta t L} . \tag{4.4}$$

Where

$$L = L^E + L^I \tag{4.5a}$$

This Lagrangian is the key to SMNA and thereby the SMNC. The Lagrangian L^G, where $G = E$ or I, is calculated using

$$L^G = \frac{(\dot{M}^G - g^G)\, g_{GG'}\, (\dot{M}^{G'} - g^G)}{2N} + \frac{M^G J_G}{2N\tau} - V' \tag{4.5b}$$

where

$$\dot{M}^G = \left[M^G(t + \Delta t) - M^G(t) \right] / \Delta t \ . \tag{4.6}$$

The function g^G in (4.5b) is equivalent to the drift factor, $K(q)$ from the single variable case discussed in Chapter II, and $g^{GG'}$ is the equivalent of $Q(q)$, or the diffusion.

$$g = det\,(g_{GG'}) \tag{4.7}$$

$$g^{GG'} = (g_{GG'})^{-1} = \tau^{-1} N^G \operatorname{sech}^2 F^G \delta_G^{G'} \ . \tag{4.8}$$

g^G function is found to be

$$g^G = -\tau^{-1}(M^G + N^G \tanh F^G) \ . \tag{4.9}$$

It is now possible to produce values for g^G and $g^{GG'}$ with the value of the intermediate variables F^G from

$$F^G = \frac{\left[V^G - \sum_{G'} a_{G'}^G \nu_{G'}^G N^{G'} - \sum_{G'} (1/2) A_{G'}^G \nu_{G'}^G M^{G'} \right]}{\left[\pi \sum_{G'} [(\nu_{G'}^G)^2 + (\phi_{G'}^G)^2](a_{G'}^G N^{G'} + (1/2) A_{G'}^G M^{G'}) \right]^{1/2}} \tag{4.10}$$

where

$$a_{G'}^G = (1/2) A_{G'}^G + B_{G'}^G \ . \tag{4.11}$$

3. Mesoscopic Parametric Values

$A^{G}_{G'}$ and $B^{G}_{G'}$ are minicolumnar averaged link efficacies and background noise, respectively, and since they are already mesoscopic columnar averages, they can be taken directly from Figure 4.2 [2]. In the case of the B^{G} values, a further modification is required before useful values can be entered into the SMNC, that is, the centering mechanism which is discussed in Appendix C must be applied. After this simple modification has been made to the $B^{G}_{G'}$ values, the weighting and background noise parameters are ready for use.

$v^{G}_{G'}$ represents minicolumnar averaged contributions to link electric polarization. $v^{G}_{G'}$ is the mean and the variance is $\phi^{G}_{G'}$. The number of parameters may be reduced using $v^{G}_{G'} = v^{G} \approx 0.1mV$ and $\phi^{G}_{G'} = \phi^{G} \approx 0.1mV$ without significantly altering the conclusions of the model [2].

The Lagrange multipliers J_G are used to represent input from interactions outside the macrocolumn. Suitable values for this parameter must be determined experimentally for each application. In the single cellular case, J_G is set equal to zero.

The equation for the mesocolumnar weighting function V'^{G} is

$$V'^{G} = \sum_{G} V''^{G}_{G'} (\rho \nabla M^{G'})^2 \tag{4.12}$$

where ρ is the spatial extent of the mesocolumn (about 0.1 mm). This equation resulted from the nearest-neighbor approximation discussed earlier, but its derivation and computer implementation are beyond the scope of this work [2]. In the SMNC, $V''^{G}_{G'}$ values are set to zero since the influence of neighboring units is not felt in the single spatial cell case.

4. Mesoscopic Variables

Equation 4.4 contains $N \approx 110$, the number of units in a mesocolumn and, τ, the mesoscopic relaxation time which is of the same order as the individual unit relaxation time $\tau \approx 5-10$ msec. As in the microscopic case, the mesoscopic interactions are modeled synchronously with $\Delta t \leq \tau$.

M^E is the number of excitatory firing units in the mesocolumn and M^I is the number of inhibitory firing units. During the operation of the SMNC, $M^G(\tau+\Delta t)$ will be calculated by the SMNA using the value of $M^G(\tau)$ from the previous iteration of the SMNC. Trial values for these parameters are found by a call to a Cauchy random number generator [36] as in the single variable cases discussed in Chapter III. The Cauchy distribution was chosen because it is a less sharply peaked distribution than the Gaussian and has a fatter tail, that is, the distribution includes "outliers" far from the mean. This feature of the Cauchy distribution ensures that the entire sample space is sampled and all minima can be found. $M(\tau+\Delta t)$ is centered around $M(\tau)$, but we must allow for values of $M(\tau+\Delta t)$ that fall far from $M(\tau)$. Szu [38] discusses the use of a Cauchy distribution in similar circumstances with respect to simulated annealing and the Lagrangian function.

The Cauchy generator returns a set of variables M^G which are input as test values to the set of equations 4.4 - 4.12. A test P is calculated and the Boltzmann method of rejection (discussed in Chapter III) is applied.

Combination	A_{jk}	B_{jk}
E-E	0.005	0.001
E-I	0.01	0.002
I-I	0.001	0.002
I-E	0.01	0.002

Figure 4.1 Microscopic Weighting And Background Noise

Combination	$A^{G}_{G'}$	$B^{G}_{G'}$
E-E	5.0	1.0
E-I	10.0	2.0
I-I	1.0	2.0
I-E	10.0	2.0

Figure 4.2 Mesoscopic Weighting And Background Noise

V. CONCLUSIONS AND RECOMMENDATIONS

This thesis research is part of a larger ongoing research effort headed by Professor Ingber at the Naval Postgraduate School. The chief contribution of this thesis has been to validate the statistical mechanical neural algorithm (SMNA) and develop the statistical mechanical neural computer (SMNC) built around the algorithm. This validation of the SMNA has prompted other researchers to encode the algorithm in the Fortran programming language and extend it to two and more variables and begin experimentation with simulations of the neocortex region of the brain and tank battles. Although the results of this thesis are interesting in their own right, their full potential cannot be understood or appreciated outside the framework of the overall research effort.

A. THE NEURAL COMPUTER

Chapter II contained a review of the history of neural computers and the current state of research in the field. From this review, it is apparent that neural computers have the potential for significant contributions in many different areas of study. Their ability to perform many simple calculations in parallel gives them the capability to find quick approximate solutions to many problems that require tremendous amounts of digital computer resources. Pattern recognition, artificial intelligence, large data base searching, analog signal processing and decision making based on incomplete data are all capabilities that have been attributed to neural computers [11]. Unfortunately, neural computing systems have been limited by their ability to manipulate the huge connection matrix that must exist to interconnect the individual computing units.

The SMNA has the demonstrated ability to produce long-time nonlinear probability distributions in systems of one variable and researchers are currently experimenting with extensions of the algorithm to systems of two and more variables. The statistical mechanical neural computer is capable of integrating the effects of adjacent temporal and spatial cells on the state of system variables through utilization of the SMNA and two statistical mechanical shortcuts discussed in Chapter IV. These shortcuts are the aggregation of about 10^2 individual units into mesocolumns, and the use of the nearest neighbor algorithm. These two concepts permit the SMNC to simulate the effects of a neural computer with about 10^{-5} as many calculations as normally would be required and overcome the chief limitation of neural computers.

B. THE SMNA

The statistical mechanical neural algorithm is a Cauchy-driven Monte Carlo calculation of the path integral solution to nonlinear Fokker–Planck equations. It provides a coarse grained approximation of the long-time probability distribution of highly complex nonlinear systems quickly and in a very memory efficient manner.

Chapter III discussed the validation of the SMNA through duplication of the results achieved by Wehner and Wolfer [6] on two different systems of single variable nonlinear equations. Particularly noteworthy was the SMNA's ability to find the two "minima" in the case of the bifurcating function and it's sensitivity to a change in initial conditions form 0.0 to 0.6 in the Rayleigh gas model. The SMNA results are coarse and have poor resolution, but this does not detract from their utility. There are many scenarios in which a commander would benefit from a quick, calculation that shows the future effect of how the variation of a force parameter affects the outcome of a battle.

The efficient use of memory is one of the chief benefits of the SMNA. The most significant memory requirement is an array that stores the values of the variables and the values of $K(q)$ and $Q(q)$. This array need only contain as many elements as N where $N = t/\Delta t$; Δt is the size of the time slice and t is the length of time the SMNA is to simulate.

Since the SMNA is memory efficient, it can easily be used on a machine with the capabilities of an IBM PC-AT microcomputer. In such an application, the Lagrangians would be computed in advance using Ingber and Upton's technique and stored on disk. The system operator would select the appropriate Lagrangian or weighted sum of Lagrangians to fit the scenario at hand. The SMNA would then quickly display the projected outcome of the battle based upon initial conditions entered interactively by the operator. Thus, the battlefield commander can have a valuable decision aid that can predict the long term consequences of his decisions regarding the combat parameters under his control.

C. THE SMNC

Due to time limitations, research was not completed on the SMNC; however, several successful runs were completed on the neocortex parameters described in Chapter IV. In one of these preliminary runs, the SMNC was operated in the single cell mode to validate the operation of the SMNA on the neocortex parameters. The objective was to reproduce Ingber's results [2] as shown in Figure 5.1a in which system minima are found for the Lagrangian L using M^E and M^I as system variables. Figure 5.1b depicts the SMNC's ability to approximate those results.

Another preliminary attempt to operate the 1089–cell two variable neocortex simulation was conducted on the NASA Ames Research Center's VAX 8800 supercomputer. The computer required approximately 88 minutes of CPU time to initialize the variables in the 1089 mesocolumnar cells and the 550 microscopic scale cells and their attendant interconnection matrices. The vast majority of the CPU time was used in initializing the microscopic scale which was designed to act as feedback and control for the mesoscopic scale. Unfortunately, delays in debugging the SMNA algorithm precluded further operational runs of the SMNC on the neocortex simulation.

This combination of an efficient engine, that is, the SMNC driving the SMNA offers tremendous potential for quick approximate solutions to many different large scale complex nonlinear systems. Systems that may eventually benefit from this approach are military C^3 systems, thermodynamics, fluid dynamics, quantum mechanics, neurobiology, lasers and perhaps even economic and political systems.

D. RECOMMENDATIONS FOR FURTHER RESEARCH

This thesis research represents one small brick in the foundation of the SMNI research project being conducted at NPS under the guidance of Professor Ingber. Many tantalizing and potentially fruitful topics have been discovered during the course of this work and others researchers have already begun to consider them. The paragraphs that follow touch on some of the more interesting topics that were discovered.

1. The Neocortex Simulation

The original intention of this research was to fully explore the SMNC's ability to approximate the results of a fully connected neural network [39]. A reasonable effort to complete the debugging of the SMNC neocortex code and perform experimental runs

would produce answers to questions such as:

- How well does the mesoscopic scale approximate the microscopic scale?
- Can the SMNC be used to study the short-term memory phenomena?
- How much can the simulated 10^5 cell network learn?
- How are learned patterns stored in a neural computer?
- Can the mesoscopic scale effectively filter the microscopic data?
- How do externally applied "commands" effect individual cells?
- Can the SMNC recognize learned patterns?

An operating simulation of the neocortex that represents about 1 mm^2 of the brain offers exciting potential for serious research applications in the field of neurobiology. Once the simulation is validated perhaps by reproducing the short-term memory phenomena, the system could be used to perform sensitivity analysis of the effect that various chemical or electrical conditions might have on the mind.

2. JANUS

In his NPS Master's thesis, Upton [7] demonstrated the capability of fitting Lagrangians to data from a simple land battle scenario modeled by the JANUS computer simulation of combat developed at Lawrence Livermore National Laboratory [40, 41]. He analyzed data from 20 JANUS battles and was able to fit the data to a Lagrangian equation of two variables. This represents the first time a complex nonlinear mathematical expression of probability has been derived from combat data.

Upton produced short-time probability distributions directly from data, and his technique is equally valid for data from computer simulations and actual combat data.

He chose a simple JANUS scenario in which two forces, Red and Blue, battled on a featureless terrain, and the variables of interest are merely the numbers of units in each force after the battle. However, the procedures used by both Upton and the SMNA will prove to be useful for scenarios of arbitrary complexity.

Ingber has coded several candidate Lagrangians into the SMNC and begun experimenting with the tank battle simulation. Further research with the SMNC operating on a multicellular terrain grid with the Lagrangian derived from JANUS data could produce answers to questions like:

- What effect do individual weapons performance parameters have on the outcome of battles?
- How similar in form are Lagrangians from computer simulations to Lagrangians that are derived from actual combat data?
- How valid are JANUS and other computer simulations?
- What is the effect of initial conditions on the outcome of battles?
- What is the effect of terrain on battles?
- Can the SMNC recognize an unfolding scenario and "anticipate" the likely enemy actions?

If tank battle simulations are interesting then surely simulated battles at sea are worthy of study, for example, outer air battles engaged in defense of a battle group. The process of fitting Lagrangians to data is one which could easily be adapted to war at sea scenarios. Such data could be found in war game centers, actual fleet exercise reports, or computer simulations. Once the Lagrangians are available, the SMNC can bring its potential to bear on an entirely new set of questions pertinent to the Navy.

3. The Battle Manager

One future potential use for the SMNC is the computerized battle manager. Such a battle manager uses the SMNC to filter the tremendous volume of data that occurs during battle to isolate patterns. The manager next compares the unfolding scenario with its memory of stored Lagrangians to determine whether it recognizes the scenario. If no recognition occurs, the battle manager will synthesize a Lagrangian from a weighted sum of such equations in memory.

Once the battle manager selects or computes the correct Lagrangian it will run a series of trajectories and examine the long-time probability distributions to anticipate the probable outcome of the battle. Given the added capability that parallel processing would bring to the SMNC, many trajectories and many variables could be examined in near real-time to find out what the future effect of a command decision might be. The human commander would assess the probabilities displayed by the battle manager and make his battle decisions accordingly.

Another potential C^3 application for the SMNC is its applicability as a force selection and weapons procurement decision aid. Procurement decisions would be based on the kind of sensitivity analysis that the SMNC could perform by individually examining each weapon parameter and variable in realistic combat scenarios. The use of the SMNC as part of such a decision aid could help ensure the maximum mix of weapons capabilities would always be available.

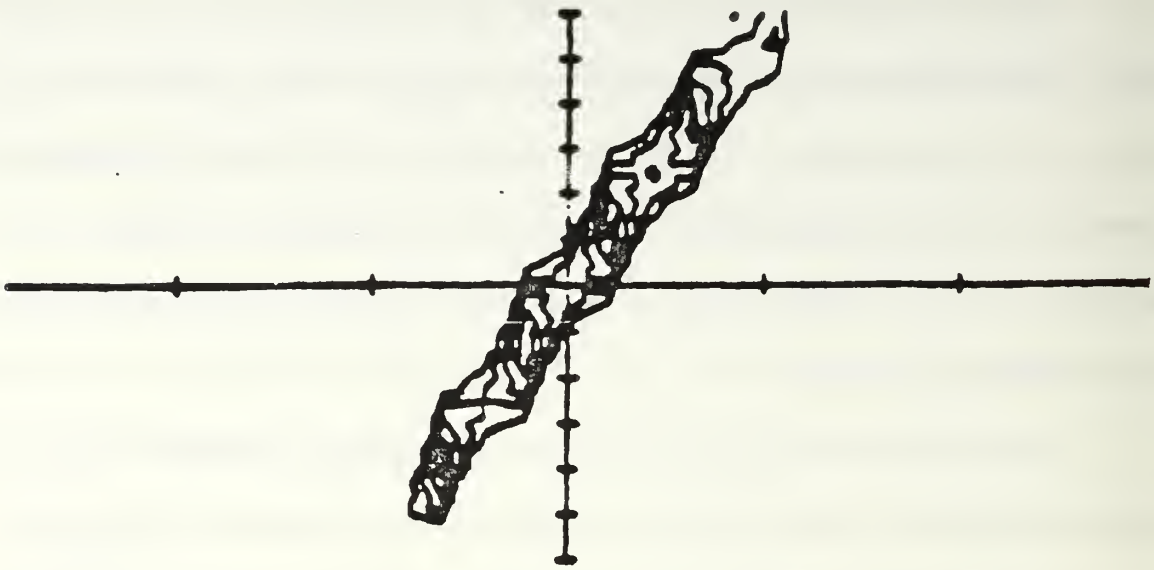

Figure 5.1a Ingber's FIG 2(b) Plot of L values < 0.04 [2]

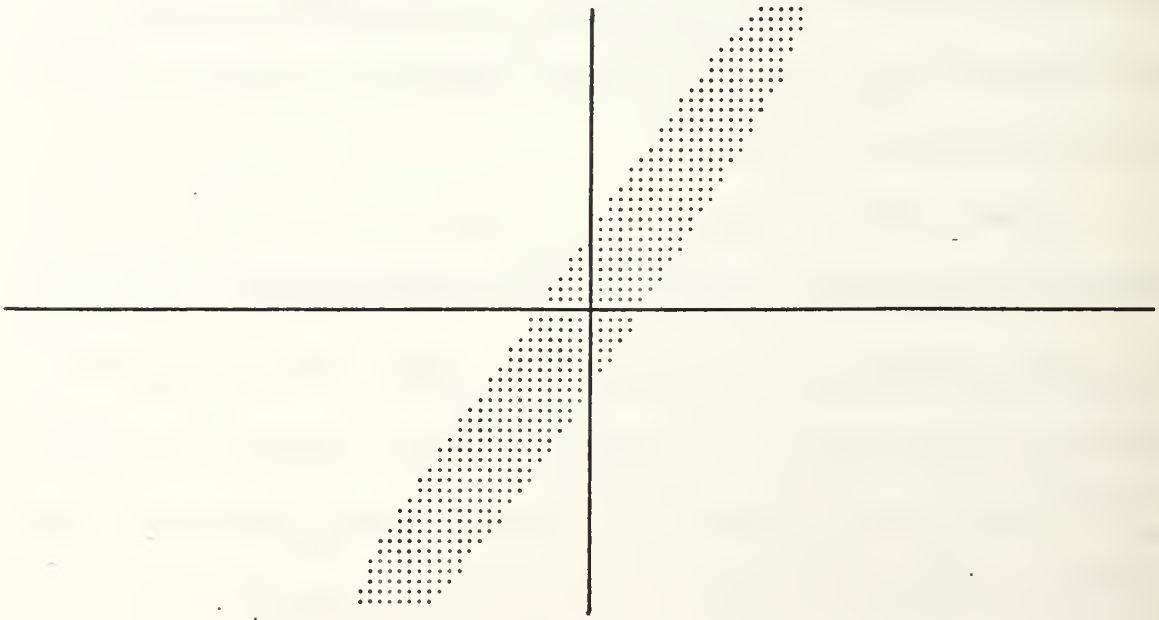

Figure 5.1b SMNC Plot of L values < 0.04

APPENDIX A: STOCHASTIC ROUTINES

All of the stochastic procedures call *get_randf* which returns a "shuffled" pseudo random floating point number uniformly distributed between 0.0 and 1.0. When first called, *get_randf* repeatedly calls a built-in random number generator which returns integers between 0 and $2^{31}-1$ (the maximum integer for the VAX 11/785). The *get_randf* procedure as shown in Figure A.1 converts the integers to floating point numbers and stores them in 256 different bins. When *get_randf* is called subsequently, it randomly selects a bin, returns the floating point number found there, and then refills the bin with a new variant. This "shuffling" technique provides an additional layer of protection against repeating sequences of numbers which might tend to lock the SMNC into artificially induced patterns of operation. This study could not afford the time to exhaustively test the "randomness" of variants returned by *get_randf*; however one simple test was performed in which a large number (100,000) of *get_randf* generated variants were analyzed to find their mean and variance. Using ten runs with different initial random number seeds, *get_randf* was found to produce variants with a mean of 0.5001 and a variance of 0.0001 which was as expected.

Uniform, shown in Figure A.2, returns uniformly distributed pseudo random integers within a range selected in the calling routine; it makes use of the shuffling provided by *get_randf*. The Gaussian procedure shown in Figure A.3 was taken from Fishman [42] and follows textbook techniques for generating Gaussian variants from uniform variants. The Gaussian procedure was tested by comparing the mean and variance of 100,000 variants generated by *gauss* with the entering mean and variance arguments. After 10

runs using different random number seeds, the difference between the entering arguments and resulting means and variances was found to be on the order of 0.0001. The runs were repeated for means of 0.001, 0.01, 0.1, 1.0 and 10.0. In each case a variance of 15% of the mean was used. Histograms of standardized variants were also examined to ensure that *gauss* returned variants possessing the classic "bell shaped" curve.

The most useful property of the Cauchy distribution for SMNC is the fact that the Cauchy distribution requires $\pm\infty$ for the upper and lower limits of variants. The testing of *cauchy*, shown in Figure A.4, involved the use of a series of histograms of standardized variants generated by repeated calls to *cauchy* with different entering median and temperature arguments. The histograms possessed the correct shapes and included variants far from the median as required of this distribution.

```
float get_randf()
   {
   static int flag;
   static float f_bin[SHUFFLE];
   unsigned int i;
   int ran_dex;
   float ret_num;
   if (flag == 0) {
      srand(SEED);
      for (i = 0; i < 512; i ++)
      rand();
      for (i = 0; i < SHUFFLE; i++)
      f_bin[i] = (float)rand() / INTMAX;
      flag = 1;
      }
   ran_dex = rand() % SHUFFLE;
   ret_num = f_bin[ran_dex];
   f_bin[ran_dex]  = (float)rand() / INTMAX;
   return(ret_num);
   }
```

Figure A.1 Get_randf( )

```
float uniform(low_bound, hi_bound)
float low_bound, hi_bound;
   {
   return(low_bound + (hi_bound - low_bound) * get_randf());
   }
```

Figure A.2 Uniform( )

```
float gauss(mu, var)
float mu, var;
   {
   double exp(), sqrt(), log();
   float get_randf();
   return(  mu + sqrt( -2 * var * log(get_randf())) *
   cos(2 * PI * get_randf()));
   }
```

Figure A.3 Gauss( )

```
float cauchy(median, temp)
float median, temp;
   {
   float get_randf();
   return(temp * tan(PI*(get_randf() - 0.5)) + median);
   }
```

Figure A.4 Cauchy( )

APPENDIX B: SINGLE VARIABLE CODE

The three functions in Figure B.1 are self explanatory. In a single variable problem, Get_K( ) and Get_Q( ) must be modified to reflect the functional form of the new Lagrangian. In a problem with two or more variables, these functions must return the elements of a matrix, the size of which is determined by the number of variables. For example, in a two variable problem such as the neocortex model, Get_K( ) becomes two functions, Get_KE( ) and Get_KI( ) and Get_Q( ) becomes Get_QE( ) and Get_QI( ). Although the code becomes somewhat more complicated, the algorithm is unchanged and effective for an arbitrary number of variables. The number of variables has no effect beyond the Get_L( ) procedures with the exception of displaying the results.

As discussed in Chapter III, the initialization procedure sets the q_0's along the zeroth trajectory to some initial value and then performs a warmup of several thousand cycles to produce a "likely" initial trajectory. This likely trajectory then becomes the initial trajectory for the subsequent update cycles. Figure B.2 shows the Init( ) procedure in which the value of the constant INIT is entered for q_0. Init( ) also computes and stores values for $K(q_0)$ and $Q(q_0)$ for later use. In a two variable problem, four values are stored; $KG(q_0)$ and $QG(q_0)$, where G is E or I.

The Update( ) procedure performs the key operations of the SMNC. Figure B.3 is the Update( ) procedure for the single variable, single cell system used to validate the SMNA. The procedure loops from 1 to NUM - 1 and provides updates to all but the final iteration or NUMth cycle. Both L2 and L4 use i+1 as an index value and consequently cannot be called when i = NUM, since NUM + 1 has no meaning. This final cycle is a

boundary case and is handled slightly differently. When i = NUM, DL = L3 - L1 rather than L4 + L3 - L1 - L2 + (log(test_Q / trajectory[i].Q) / 2). It is this final iteration that produces the values of q for plotting.

Figure B.4 is the main program which contains the display algorithms. For the sake of convenience during the debugging and operating of the code, constants such as H, V, RESOLUTION, and the like were set globally in the declaration part of the code. Thus manipulation of parameters could be done at one place in the program code, namely the beginning.

```
float get_K(q)
float q;
 {
 return(tanh(q));
 }

float get_Q(q)
float q;
 {
 return(1.0);
 }

float get_L(q1, q2, K, Q)
float q1, q2, K, Q;
 {
 return((q1 - q2 - (K * DELTA_T)) * (q1 - q2 - (K * DELTA_T)) /
  (2.0 * Q * DELTA_T));
 }
```

Figure B.1 Get_K( ), Get_Q( ) and Get_L( )

```
init( )
 {
 unsigned int i;
 trajectory[0].q = INIT;
 trajectory[0].K = get_K(trajectory[0].q);
 trajectory[0].Q = get_Q(trajectory[0].q);
 for(i = 1; i < NUM1; i ++) {
  trajectory[i].q = INIT;
  trajectory[i].K = get_K(trajectory[i].q);
  trajectory[i].Q = get_Q(trajectory[i].q);
  }
 for(i = 0; i < WARMUP; i++)
  update();
 }
```

Figure B.2 Init( )

```
update()
 {
 unsigned int i;
 int flag_q;
 float DL, L1, L2, L3, L4, q_prime, test_K, test_Q, x, y, cauch;
 cauch = CAUCHY;
 for (i = 1; i < NUM; i++) {
  L1 = get_L(trajectory[i].q, trajectory[i-1].q,
    trajectory[i-1].K, trajectory[i-1].Q);
  L2 = get_L(trajectory[i+1].q, trajectory[i].q,
    trajectory[i].K, trajectory[i].Q);
  q_prime = cauchy(trajectory[i].q, temp);
  while((q_prime <= -cauch) || (q_prime > cauch))
    q_prime = cauchy(trajectory[i].q, temp);
  test_K = get_K(q_prime);
  test_Q = get_Q(q_prime);
  L3 = get_L(q_prime, trajectory[i-1].q,
    trajectory[i-1].K, trajectory[i-1].Q);
  L4 = get_L(trajectory[i+1].q, q_prime,
    test_K, test_Q);
  DL =  L4 + L3 - L1 - L2 + (log(test_Q / trajectory[i].Q) / 2);
  flag_q = 0;
  if(DL < -25.0) {
    x = 1.0;
    y = 0.0;
    }
  else if(DL > 25.0)  {
    x = 0.0;
    y = 1.0;
    }
  else {
    x = exp(-DL);
    y = get_randf();
    }
  if(x > y)
    flag_q = 1;
  if(flag_q == 1) {
    trajectory[i].q = q_prime;
    trajectory[i].K = test_K;
    trajectory[i].Q = test_Q;
    }
 }
```

Figure B.3 Update( )

```
main ()
 {
 unsigned int i, j;
 int hist[60];
 float n_bin[60], q_scale, ratio, factor,  height;
 ratio = Y_INCHES / X_INCHES;
 q_scale = (1.0 / RESOLUTION)  * (1.0 / H);
 factor = q_scale * V * ratio;
 init();
 for (i = 0; i < N; i++) {
  if(i % 1000 == 0)
   init();
  update();
  hist[30 + (int)(trajectory[NUM].q)] += 1;
  }
 for(i = 0; i < (SCALE / RESOLUTION); i ++) {
  printf("593d",  i - 30);
  n_bin[i] = (float)(hist[i] / (float)N);
  height = n_bin[i] * factor  / RESOLUTION;
  for(j = 0; j < height; j++)
   printf(" ");
  printf("x");
  }
 }
```

Figure B.4 Main( )

APPENDIX C: CENTERING MECHANISM

The Lagrangian L is one of the critical values in the mesoscopic scale. It leads to to the appropriate choices of the M^G values through a Boltzmann type method of rejection. Additionally, L can provide information about the minima of M^G which permits the application of a centering mechanism.

Examine the following equation:

$$F^G = \frac{\left[V^G - \sum_{G'} a^G_{G'} v^G_{G'} N^{G'} - \sum_{G'} (1/2) A^G_{G'} v^G_{G'} M^{G'} \right]}{\left[\pi \sum_{G'} [(v^G_{G'})^2 + (\phi^G_{G'})^2] (a^G_{G'} N^{G'} + (1/2) A^G_{G'} M^{G'}) \right]^{1/2}}, \tag{C.1}$$

It has been shown [2] that more minima of L are found when the numerator contains terms only in M^G. In other words, the constant term = 0 in the numerator. Statistically, any mechanism that promotes more or deeper minima is to be favored and Ingber [2] has shown that a centering mechanism has plausible support in the neocortex. Since a centering mechanism is therefore considered desirable in the SMNC, the only remaining questions concern implementation.

Actually, for all the benefits to be derived in the SMNC from the use of a centering mechanism, the implementation is almost trivially easy. It is accomplished simply by adjusting the synaptic background noise factors B^G_E to B'^G_E. This is done by solving

$$V^E - \left[((1/2) A^E_E + B^E_E) v^E_E N^E + ((1/2) A^E_I + B^E_I) v^E_I N^I \right] = 0 \tag{C.2}$$

for both $G=E$ and $G=I$.

Care must be taken to discard results that represent the non-physical case of $B'^{G}_{E}<0$ in the square root of the denominator of Eq. 1. Fortunately, in all cases where a negative B'^{G}_{E} occurs, it is possible to find a positive value for B'^{G}_{I}. In addition to eliminating the constant term in the numerator of Eq. 1, the centering mechanism also modifies the constant terms in the denominator of Eq. 1.

The overall effect of implementing the centering mechanism is to increase the number of minima of L and cluster the minima about zero. The SMNC operates in the dominant excitation environment with the centering mechanism in operation. It is a trivial task to alter the parameters to change environments to dominant inhibition or balanced.

LIST OF REFERENCES

[1] Ingber, L., "Towards a unified brain theory," *J. Social Biol. Struct. 4*, 211-224 (1981).

[2] Ingber, L., "Statistical mechanics of neocortical interactions. Derivation of short-term-memory capacity," *Phys. Rev. A 29*, 3346-3358 (1984).

[3] Ingber, L., "Statistical mechanics of neocortical interactions. I. basic formulation," *Physica D 5*, 83-107 (1982).

[4] Ingber, L., "Statistical mechanics of neocortical interactions: Stability and duration of the 7±2 rule of short-term-memory capacity," *Phys. Rev. A 31*, 1183-1186 (1985).

[5] Haken, H., *Synergetics*, 3rd ed. (Springer, New York, 1983).

[6] Wehner, M. F. and Wolfer, W. G., "Numerical evaluation of path-integral solutions to Fokker-Planck equations," *Phys. Rev. A 27*, 2663-2670 (1983).

[7] Upton, S., "A Statistical Mechanics Model of Combat," Masters Thesis, Naval Postgraduate School, Monterey, CA, March 1987.

[8] *Letter 3910 RPDP-3-89, Ser 10P* (Office of Naval Research, Washington, DC, 23 Dec 1986).

[9] Abu-Mustafa, Y. F. and Psaltis, D., "Optical neural computers," *Scientific American* 88-96 (March 1987).

[10] Green, L., "Neural-net systems: Computers that learn," *Information Week* 32 (16 March 1987).

[11] Port, O., "Computers that come awfully close to thinking," *Business Week* 92-97 (2 June 1986).

[12] Brown, C., "Parallel optical computer solves multivariable problems in realtime," *Engineering Times* 3-4 (1 Dec 1986).

[13] Williams, T., "Optics and neural nets: Trying to model the human brain," *Computer Design* 47-62 (1 March 1987).

[14] Brown, R. J., "An artificial neural network experiment," *Dr. Dobb's J. of Software Tools* 16-27 (April 1987).

[15] Kinoshita, J. and Palevsky, N. G., "Computing with neural networks," *High Technology* (May 1987).

[16] Minsky, M. and Papert, S., *Perceptrons* (MIT Press, Cambridge, MA, 1965).

[17] Grossberg, S., "The quantized geometry of visual space: The coherent computation of depth, form, and lightness," *Behav. Brain Sci. 6*, 625-692 (1983).

[18] Grossberg, S., "Associative and Competitive Principles of Learning and Development," in *Competition and Cooperation in Neural Nets*, ed. by S. Amari and M.A. Arbib (Springer, New York, 1982).

[19] Stevens, C. F., "The Neuron," in *The Brain 'A Scientific American Book'*, ed. by D. Flanagan et al (W. H. Freeman, New York, 1979).

[20] Sampson, J. R., *Adaptive Information Processing* (Springer, New York, 1976).

[21] Clark, J. W., Rafelski, J., and Winston, J. V., "Brain without mind: Computer simulation of neural networks with modifiable neural interactions," *Phy. Rep. 123*, 215-273 (1985).

[22] Rosenblatt, F., *Principles of Neurodynamics* (Spartan, Washington, DC, 1961).

[23] Hawkins, J. K., "Self-organizing systems -- A review and commentary," *Proc. IRE* 31-48 (Jan 1961).

[24] Peretto, P. and Niez, J., "Stochastic dynamics of neural networks," *IEEE Trans. Syst. Man Cyber. SMC-16*, 73-83 (1986).

[25] Hopfield, J. and Tank, D. W., "Computing with neural circuits: A model," *Science 233*, 625-633 (1986).

[26] Hecht-Nielsen, R., *Performance Limits of Optical, Electro-Optical and Electronic Neurocomputers* (Hecht-Nielsen Neurocomputer Corp., San Diego, CA, 1987).

[27] "Introduction to Data Level Parallelism," Thinking Machine Technical Report 86.14, Cambridge, MA, 1986.

[28] *Transputer Architecture, Technical Overview* (INMOS Corp., Colorado Springs, CO, 1985).

[29] Ingber, L. and Upton, S., "Stochastic model of combat ," in *1987 Symposium on C3 Research*, ed. by M. Sovereign (National Defense University, Washington, D.C., 1987).

[30] Dupuy, T. N., "Can we rely on computer combat simulations," *Armed Forces J.* (August 1987).

[31] Schulman, L. S., Techniques and Applications of Path Integration, New York, 1981.

[32] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., "Equation of state calculations by fast computing machines," *j. chem. phys. 21 Nr. 6*, (June 1953).

[33] Landau, D. P. and Alben, R., "Monte Carlo calculations as an aid in teaching statistical mechanics," *AJP 41*, (1973).

[34] Szu, H., "Globally connected network models for computing using fine grained processing elements," *Proceedings of LASER-85* (1985).

[35] Ingber, L., "C3 decision aids: statistical mechanics application of biological intelligence," in *1987 Symposium on C3 Research*, ed. by M. Sovereign (National Defense University, Washington, D.C., 1987).

[36] Bratley, P., Fox, B. L., and Schrage, L. E., *A Guide to Simulation* (Springer, New York, 1983).

[37] Hubel, D. H. and Wiesel, T. N., "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Phisiol 160*, 293-332 (1962).

[38] Szu, H., "Non-convex optimization," *SPIE 698*, (to be published 1987).

[39] Connell, J., Ingber, L., and Yost, C., "Statistical Mechanical Virtual Neural Computer," in *1987 Symposium on C3 Research*, ed. by M. Sovereign (National Defense University, Washington, DC, 1987).

[40] The JANUS Manual Version, 3.02, November 1986.

[41] JANUS Algorithm Document Version, 3.01 , September 1986.

[42] Fishman, G. S., *Concepts and Methods in Discrete Event Digital Simulation* (Wiley, New York, 1973).

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor John F. Powers Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5004	1
4.	Professor Lester Ingber Physics Department Naval Postgraduate School Monterey, California 93943-5004	30
5.	Professor Carl Jones Naval Postgraduate School Monterey, California 93943-5004	5
6.	Professor K. Woehler Naval Postgraduate School Monterey, California 93943-5004	2
7.	Dean Gordon Schacher Naval Postgraduate School Monterey, California 93943-5004	1
8.	Professor Don Harrison Physics Department Naval Postgraduate School Monterey, California 93943-5004	1
9.	Maj Richard Adams, USAF Naval Postgraduate School Monterey, California 93943-5004	2

10. MAJ Bernie Galing, USA — 1
TRADOC Monterey
Monterey, California 93943-5004

11. Dr. S. Christian Simonson III — 1
Conflict Simulation Center
Evaluation and Planning Division
Lawrence Livermore National Laboratory
Livermore, California 94550

12. Dr. J. B. Adams — 1
Sandia Laboratories
Livermore, California 94550

13. Dr. William G. Wolfer — 1
Sandia Laboratories
Livermore, California 94550

14. Dr. Mike F. Wehner — 1
"B" Division
Lawrence Livermore National Laboratory
Livermore, California 94550

15. Capt Stephen C. Upton, USMC — 1
Plans D Dev
MCDEC
Quantico, Virginia 22134-5080

16. Adrian S. Yano — 1
2124 Kittredge Street
Berkeley, California 94704

17. Professor Aram Mekjian — 1
The State University of New Jersey
Rutgers
Department of Physics and Astronomy
P.O. Box 849
Piscataway, New Jersey 08854

18. Professor Andrew U. Meyer — 1
New Jersey Institute of Technology
Department of Electrical Engineering
Newark, New Jersey 07102

19. Professor Michael E. Melich — 1
Naval Research Laboratory
Code 7570
Washington, D.C. 20375

20. Professor Michael G. Sovereign
USRADCO, STC
APO New York, New York 09159

21. Dr. Harold Szu — 1
Code 5709
Naval Research Lab
Washington, DC 20375-5000

22. COL Gary Q. Coe, USA — 1
Chief, Modeling & Analysis Division (J6-F)
Office of the Joint Chiefs of Staff
The Pentagon, Room 1D827
Washington, DC 20301-5000

23. CPT Thomas Moore, USA — 1
Y Division
UC Lawrence Livermore Lab
Livermore, California 94550

24. Professor Douglas Jones — 1
School of Engineering and Applied Science
Department of Civil, Mechanical &
Environmental Engineering
George Washigton University
Washington, DC 20052

25. CDR John C. Connell, Jr., USN — 10
Defense Nuclear Agency
801 Telegraph Rd.
Alexandria, Virginia 22310

26. LCDR Charles Yost, USN — 1
VR-22
FPO New York, New York 09540

27. Dr. Henry Lum — 1
NASA Ames Research Center
Mail Stop 244-7
Moffet Field, California 94035

28. Professor Bruce McClennan — 1
Department of Computer Science
Ayres Hall
University of Tennessee
Knoxville, Tennessee 37996-1301

29. CDR Thomas E. Halwachs, USN — 1
Code 30
Naval Postgraduate School
Monterey, California 93943

30. Mike Mudurian — 1
Code 421
Naval Ocean Systems Center (NOSC)
San Diego, California 92152

31. Dr. Irwin Goodman — 1
Code 421
Naval Ocean Systems Center (NOSC)
San Diego, California 92152

32. COL Don Blumenthal USA(RET) — 1
Conflict Simulation Center
Evaluation and Planning Division
Lawrence Livermore National Laboratory
Livermore, California 94550

33. Professor Vincent Y. Lum — 2
Chairman Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5004

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

Thesis
C7045
c.1

Connell
Memory efficient evaluation of nonlinear stocastic equations and C^3 applications.

Thesis
C7045
c.1

Connell
Memory efficient evaluation of nonlinear stocastic equations and C^3 applications.

AMERICAN BINDERIES

thesC7045
Memory efficient of nonlinear stocastic
3 2768 000 77205 7
DUDLEY KNOX LIBRARY